

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

ПРЕЗЕНТАЦИЯ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

дисциплина: Математические основы защиты информации и информационной безопасности

Преподаватель: Кулябов Дмитрий Сергеевич

Студент: Попова Юлия Дмитриевна

Группа: НФИмд-01-23

МОСКВА

2023 г.

Прагматика выполнения лабораторной работы

Требуется реализовать:

1. Маршрутное шифрование.
2. Шифрование с помощью решеток.
3. Табоица Виженера

Цель работы

Освоить на практике шифры перестановки.

Выполнение лабораторной работы

1. Для реализации маршрутного шифрования:

1. Функции заполнения матрицы текстом
2. Функция, шифрующая матрицу

```
import math
import re

1 usage
def is_pass_valid(password, columns):
    if len(password) != columns:
        print("pass len should be equal to row len")
        raise ValueError
    alphabeticValues = re.findall(r'[a-zA-Za-яA-Я]', password)
    if len(alphabeticValues) != len(password):
        print("pass should contain only eng and ru letters")
        raise ValueError

1 usage
def is_len_columns_valid(columns, inputString):
    if columns > len(inputString):
        print("len columns should be <= len of Input String")
        raise ValueError
    if columns <= 1:
        raise ValueError

1 usage
def to_dict(inputString, password, columns, row):
    list_of_slices = []
    for i in range(row):
        list_of_slices.append(inputString[columns * i:columns * (i + 1)])
```

```

while len(list_of_slices[-1]) != columns:
    list_of_slices[-1] += 'a'

routeDict = {}

for i in range(columns):
    stringToAppend = ""
    for j in range(row):
        stringToAppend += list_of_slices[j][i]
    routeDict[password[i]] = stringToAppend

sorted_dict = dict(sorted(routeDict.items()))

return sorted_dict

inpututedString = input("Input string to encrypt: ")
columns = int(input("Input int value to determine the number columns: "))
password = input("Password: ").lower()

inputString = inpututedString.replace(" ", "") # remove spaces
password = password.replace(" ", "") # remove spaces
password = ''.join([password[i] for i in range(len(password) - 1) if password[i + 1] != password[i]] + [
    password[-1]]) # remove duplicates
is_len_columns_valid(columns, inputString)
is_pass_valid(password, columns)
row = math.ceil(len(inputString) / int(columns))

routeDict = to_dict(inputString, password, columns, row)

```

```

routeDict = to_dict(inputString, password, columns, row)

cryptogram = ""
for key in routeDict:
    cryptogram += routeDict[key]

print("начальная фраза: ", inpututedString)
print("криптограмма: ", cryptogram)

```

2. Запуск программы маршрутного шифрования

```
"C:\Users\Asuser\Desktop\учеба\магистратура\inf sec\lab2\Script
Input string to encrypt:  нельзя недооценивать противника
Input int value to determine the number columns: 6
Password:  пароль
начальная фраза:  нельзя недооценивать противника
криптограмма:  еенпнзоатаьовокннеевлдирияцтиа

Process finished with exit code 0
```

3. Для реализации шифрования с помощью решеток:

Чтобы реализовать программу был написан код на python:

```
1 import numpy as np
2
3 1 usage
4 def generate_grid(k):
5     grid = np.zeros((k, k))
6     for i in range(k):
7         for j in range(k):
8             grid[i][j] = i * k + j + 1
9     return grid
10
11 1 usage
12 def encrypt_with_grid(grid, text):
13     k = len(grid)
14     encrypted = np.chararray((2*k, 2*k), unicode=True)
15     encrypted[:] = ''
16
17     idx = 0
18     for _ in range(4):
19         for i in range(k):
20             for j in range(k):
21                 if grid[i][j] and idx < len(text):
22                     encrypted[i][j] = text[idx]
23                     idx += 1
24
25         grid = np.rot90(grid)
26
27     return encrypted
```

```

1 usage
27 def extract_by_password(grid, password):
28     order = sorted(range(len(password)), key=lambda x: password[x])
29     return ''.join([''.join(grid[:, i]) for i in order])
30
31 usage
32 def encrypt(text, password):
33     k = int(len(text)**0.5)
34     if k*k != len(text):
35         raise ValueError("Length of the text should be a perfect square.")
36
37     if len(password) != k:
38         raise ValueError(f"Length of the password should be {k}.")
39
40     grid = generate_grid(k)
41     encrypted_grid = encrypt_with_grid(grid, text)
42     result = extract_by_password(encrypted_grid, password)
43
44     return result
45
46 # Пример использования
47 text = "нужно подписать новый указ"
48 password = "шифр"
49 print(encrypt(text, password))

```

4. Запуск программы

```

45 #💡Пример использования
46 text = "договор подписали"
47 password = "шифр"
48 print(encrypt(text, password))

```

grid_encryption (1) ×

⋮

"C:\Users\Asuser\Desktop\учеба\магистратура\inf sec\lab2\Scripts\python.exe" "C:\oodaopиигрплдвос

Process finished with exit code 0

6. Для реализации Таблицы Виженера:

1. Функция шифрования (построение таблицы Виженера)

```
1 usage
2 def vigenere_encrypt(text, key):
3     alph = 'абвгдежзийклмнопрстуфхцчшщъыьэюя'
4     encrypted_text = ''
5
6     # размерность пароля = размерности текста
7     key_expanded = ''
8     while len(key_expanded) < len(text):
9         key_expanded += key
10    key_expanded = key_expanded[:len(text)]
11
12    # сравниваем по таблице и получаем на пересечении букву
13    for t, k in zip(text, key_expanded):
14        if t in alph:
15            new_pos = (alph.index(t) + alph.index(k)) % len(alph)
16            encrypted_text += alph[new_pos]
17        else:
18            encrypted_text += t
19
20    return encrypted_text
21
22 usage
23 def vigenere_decrypt(encrypted_text, key):
24     alph = 'абвгдежзийклмнопрстуфхцчшщъыьэюя'
25     decrypted_text = ''
```

```

25
26     key_expanded = ''
27     while len(key_expanded) < len(encrypted_text):
28         key_expanded += key
29     key_expanded = key_expanded[:len(encrypted_text)]
30
31     for e, k in zip(encrypted_text, key_expanded):
32         if e in alph:
33             new_pos = (alph.index(e) - alph.index(k)) % len(alph)
34             decrypted_text += alph[new_pos]
35         else:
36             decrypted_text += e
37
38     return decrypted_text
39
40
41 # Пример использования
42 text = "криптографиясерьезнаянаука"
43 password = "математика"
44 encrypted_text = vigenere_encrypt(text, password)
45 print(f"Encrypted: {encrypted_text}")
46 print(f"Decrypted: {vigenere_decrypt(encrypted_text, password)}")
47

```

7. Запуск программы Таблицы Виженера

```

41 # Пример использования
42 text = "криптографиясерьезнаянаука"
43 password = "математика"
44 encrypted_text = vigenere_encrypt(text, password)
45 print(f"Encrypted: {encrypted_text}")
46 print(f"Decrypted: {vigenere_decrypt(encrypted_text, password)}")

```

Run vigenere_encryption x

```

"C:\Users\Asuser\Desktop\учеба\магистратура\inf sec\lab2\Scripts\python.exe" "C:\Users\Asuser\Desktop\магистратура\lab2\scripts\vigenere_encryption.py"
Encrypted: црѣфюохшкфягкьчпчалнтшца
Decrypted: криптографиясерьезнаянаука
Process finished with exit code 0

```

Выводы

В результате выполнения работы освоили на практике применение шифров перестановки.